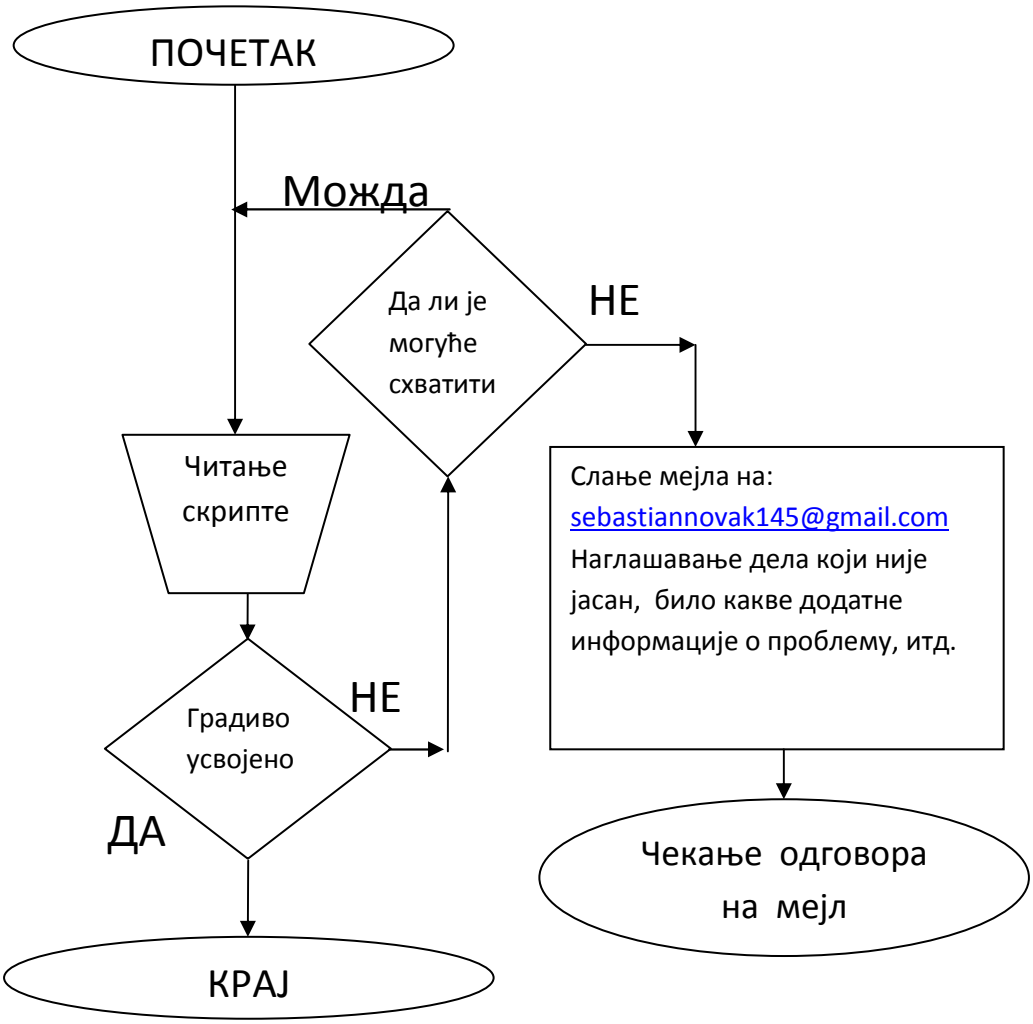
АЛГОРИТМИ

Себастиан Новак



Предговор

Скрипта је намењена ђацима трећег разреда средњих школа као увод у програмирање и као додатак предавањима аутора у школи математике и рачунара "Рајак".

Градиво је изложено поступно, са теоријским основама и детаљно објашњеним практичним примерима. У првом поглављу се постављају темељи неопходни за разумевање алгоритама и њихово формулисање. Друго поглавље се бави типовима алгоритама и типичним математичким проблемима. Треће поглавље садржи решене примере.

Аутор позива све читаоцима да своја запажања, примедбе и корекције пошаљу на адресу sebastiannovak145@gmail.com и за то им се унапред захваљује.

Аутор

Садржај

1. Алгоритми - увод

Појам алгоритма- неформално и интуитивно алгоритам схватамо као поступак. Још у IX веку Ал-Ховаризми је формулисао поступак у виду правила за извршавање основних аритметичких операција. По њему је настала реч Алгоритам.
Дакле алгоритам је **поступак** за решавање неког проблема који је формално написан али и даље **интуитивно** разумљив.
Емпиријска дефиниција алгоритма:
Алгоритам је правило, формулисан на неком језику који једнозначно дефинише редослед операција неопходних за трансформацију улазних података у тражени резултат.

1.1 Решавање проблема - алгоритамско размишљање

Када формирамо алгоритам, односно решавамо неки проблем најпре морамо **разумети проблем.** Шта би значило „разумети проблем“ ?
Па најпре налазимо шта се од нас тражи, шта нам је све на доступању од информација везаних за наш проблем и затим размишљамо о правилима на основу којих морамо решити проблем.
Након разумевања проблема, записујемо информације и идеје које тренутно имамо **природним језиком.**

Када смо успели формирати неко решење од тренутних идеја које имамо, од њега правимо **модел**, односно формалније и прецизније записујемо кораке ка решењу проблема.
На основу модела формирамо алгоритам.
Овим смо видели да при решавању неког проблема и формирању одговарајућег алгоритма имамо следеће кораке:

1. Разумевање проблема
2. Записивање на природном језику идеја за решавање проблема
3. Изградња модела (писање математичких формула, израза)
4. Формулисање алгоритма

Наш први задатак: ***Направити Фејсбук профил.***

1. Разумевање проблема

Шта значи направити Фејсбук профил?
- Отићи на www.facebook.com
- Попунити поља у формулару за регистрацију и кликнути на “Региструј се”

2. Записивање на природном језику идеја за решавање проблема

-Одлазак на www.facebook.com

-Унос имена

-Петар

-Унос презимена

- Петровић

-Унос е-маил адресе

-Питање, да ли имам е-маил адресу?

НЕМАМ – направим је, унесем је

ИМАМ - унесем је

-Унесем лозинку

-1111аааа

-Изаберем пол

Мушко

-Изаберем датум рођења

1.1.1980

-Кликнем на “Региструј се”

3. Изградња модела (писање математичких формула, израза)

Да би могли формално записали изразе, прво ћемо рећи пар ствари о појму променљиве у програмирању. Променљива је именовани простор у меморији рачунара, који је намењен чувању вредности. Именовани простор у меморији рачунара значи да ће променљива добити своје место у меморији, на ком ће писати њено име. При уносу података у програм, податке смештамо у променљиве, како би могли да их користимо у току рада програма. Тако да, када напишемо **Име**=? , ми заправо тражимо унос неке вредности за променљиву Име. Писањем **Име**=Петар додељујемо променљиви Име вредност Петар.

Унутар меморије рачунара се на месту које припада променљивој **Име** уписује вредност Петар.

Простор у меморији рачунара додељен променљиви

Име:	Петар

Остатак простора меморије рачунара, чији садржај нам није познат.

Променљивима додељујемо вредности при уноси и при извршавању програма.

Име=?
Презиме=?
е-маил=?

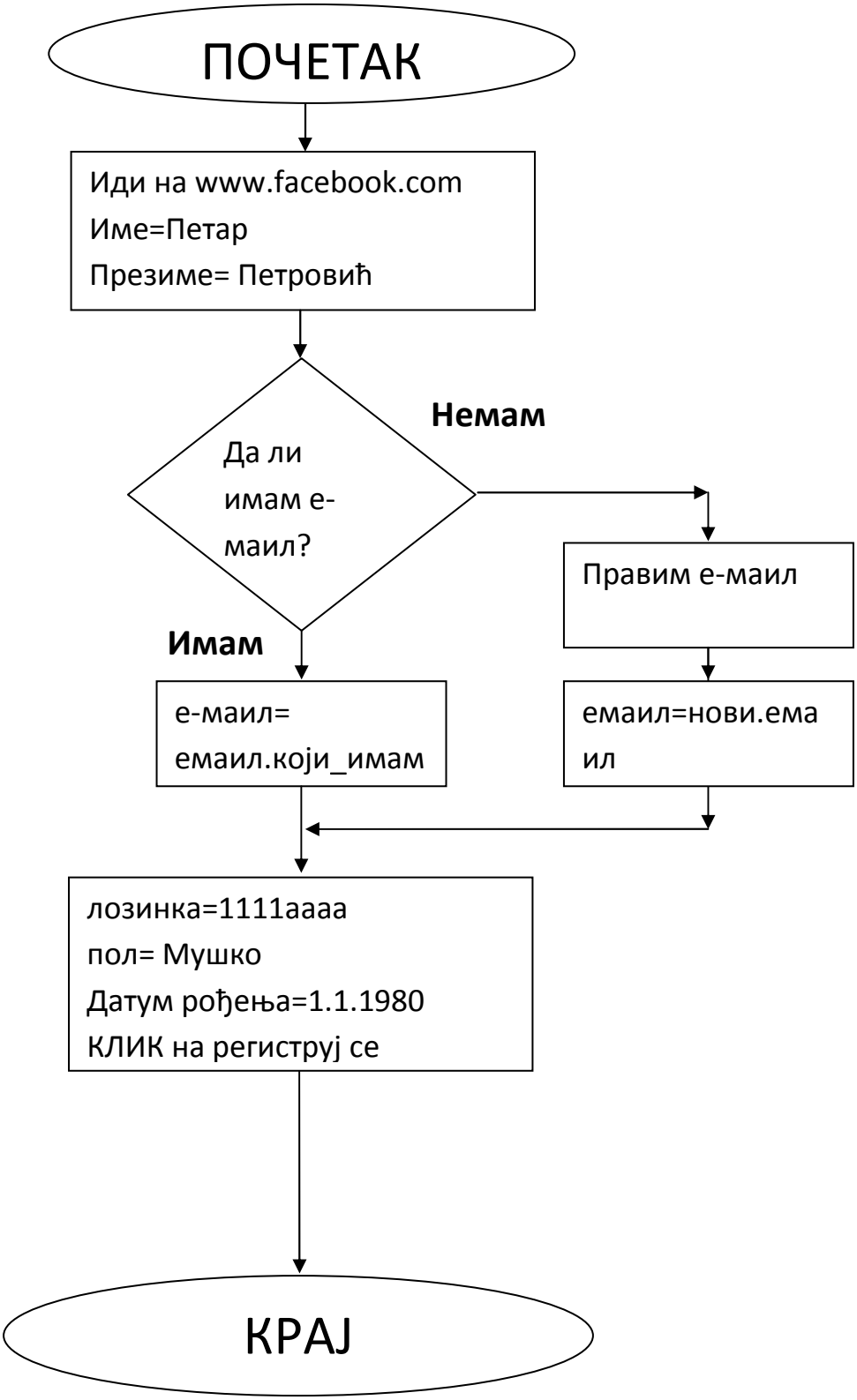
лозинка=?
пол=?
Датум рођења=?

Име=Петар
Презиме= Петровић
Да ли имам е-маил? ИМАМ/НЕМАМ
НЕМАМ

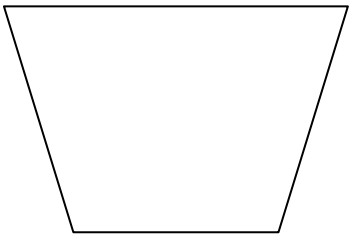
Прављење е-маил адресе
е-маил=нови.емаил
ИМАМ
е-маил=емаил.који_имам

лозинка=1111аааа
пол= Мушко
Датум рођења=1.1.1980
-Сви кораци завршени, клик на Региструј се

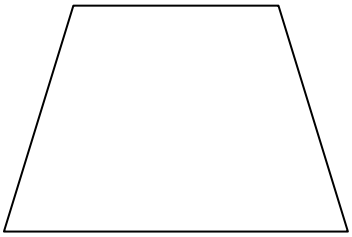
4. Формулисање алгоритма



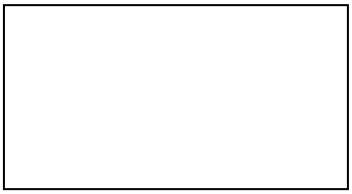
1.2 Цртање алгоритама



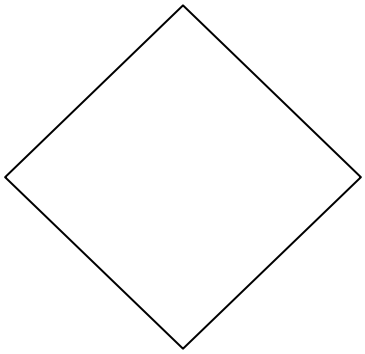
Унос података



Испис података



Израчунавање

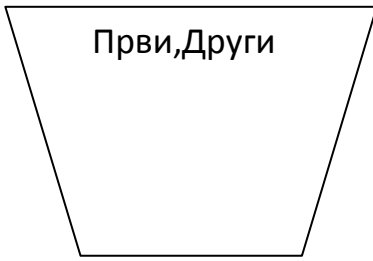


Гранање на основу услова

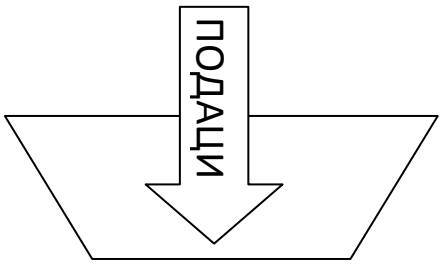
Цртање сваког алгоритма започињемо са блоком

а завршавамо са

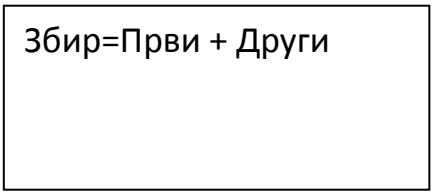
Унос података представљамо са обрнутим једнакокраким трапезом.



Стављамо га најчешће након почетка и уз помоћ њега уносимо неке податке за које нам је у тексту задатка остављене произвољне вредности. На пример ако се тражи формулисање алгоритма који сабира два унета броја, користићемо овај блок, јер алгоритам треба да сабира било која два броја, по избору корисника.

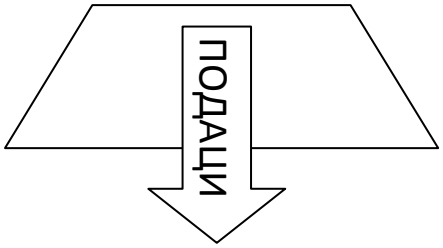


Затим морамо представити операцију сабирања два броја и за то користимо следећи блок(блок за израчунавање):

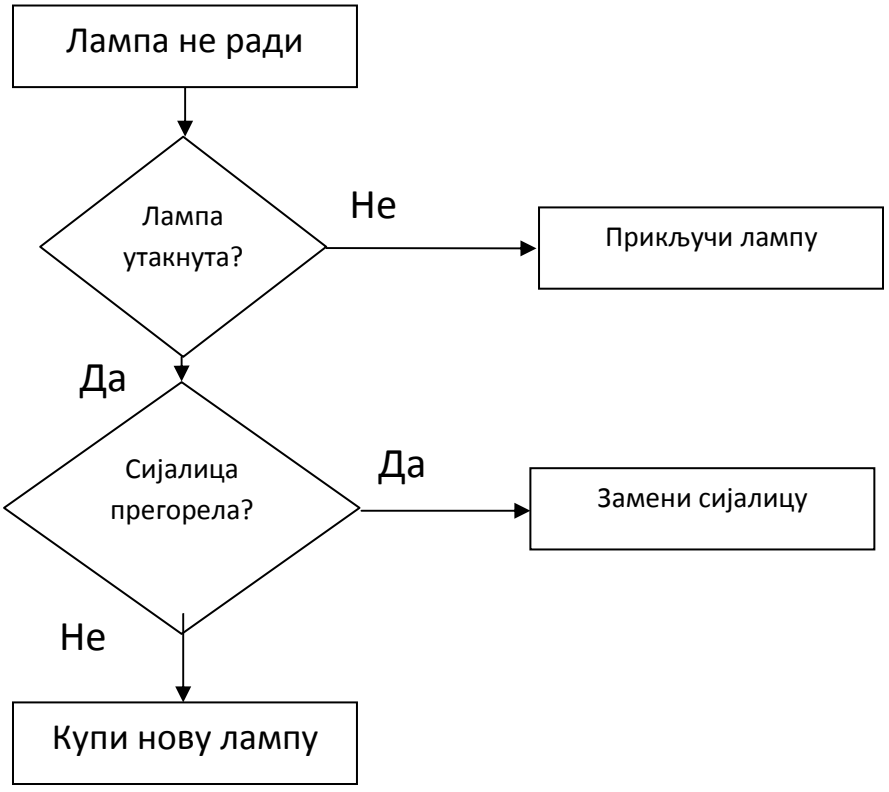


Када се та два броја саберу, резултат операције ће бити сачуван у

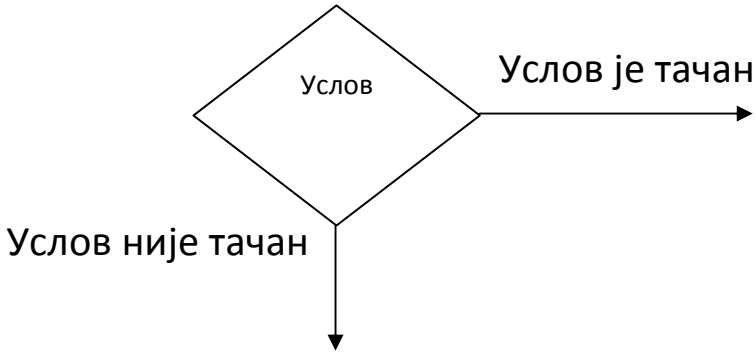
“Збир” и сада би га требало исписати односно представити кориснику на неки начин. За исписивање резултата неке аритметичке операције или обичног текста, користимо блок за испис који се представља једнакокраким трапезом, ако је у питању резултат, написаћемо име под којим смо га сачували, ако је реч о обичном тексту ставићемо га под наводнике.



Често да би решили неки проблем морамо проверити тачност разних логичких услова, на пример да би сазнали зашто нека лампа не ради морамо проверити неколико ствари (односно услова):



Блок за гранање се садржи из услова и путања које се бирају на основу тачности услова.



Осим оваквих логичких конструкција, које извршавају једну радњу, једном , уз помоћ блока за гранање можемо контруисати и циклусе, који ће се извршавати задат број пута или док услов не буде задовољен.
Ову конструкцију ћемо звати бројачки циклус, а алгоритме који се заснивају на њој звати цикличним алгоритмима.

1.3 Циклуси – петље

Постоје две врсте циклуса, они за које знамо колико ће се пута радње у њима извршити и оне за које не знамо. Врсте циклуса су:

- Бројачки циклуси – зависе од вредности бројача
- Циклуси са предусловом – зависе од логичког услова који се проверава пре уласка у циклус
- Циклуси са постусловом – зависе од логичког услова који се проверава након уласка у циклус и извршавања радње циклуса

1.3.1 Бројачки циклус

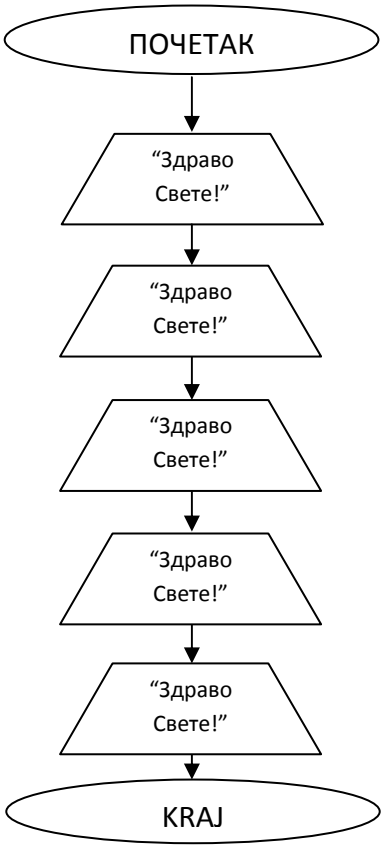
Бројачки циклус чине следеће компоненте:

- Бројач, који се најчешће на почетку програма ставља на 0 или 1, или било коју другу вредност у зависности од услова.
- Радње, израчунавања и операције које се понављају докле год је услов тачан
- Услов чија се тачност провера сваки пут и на основу ње одлучује да ли ће се радње извршити или не.
- Операције које модификује бројач (најчешће је то увећавање вредности за 1 или умањивање, али може бити и било која друга у зависности од услова)

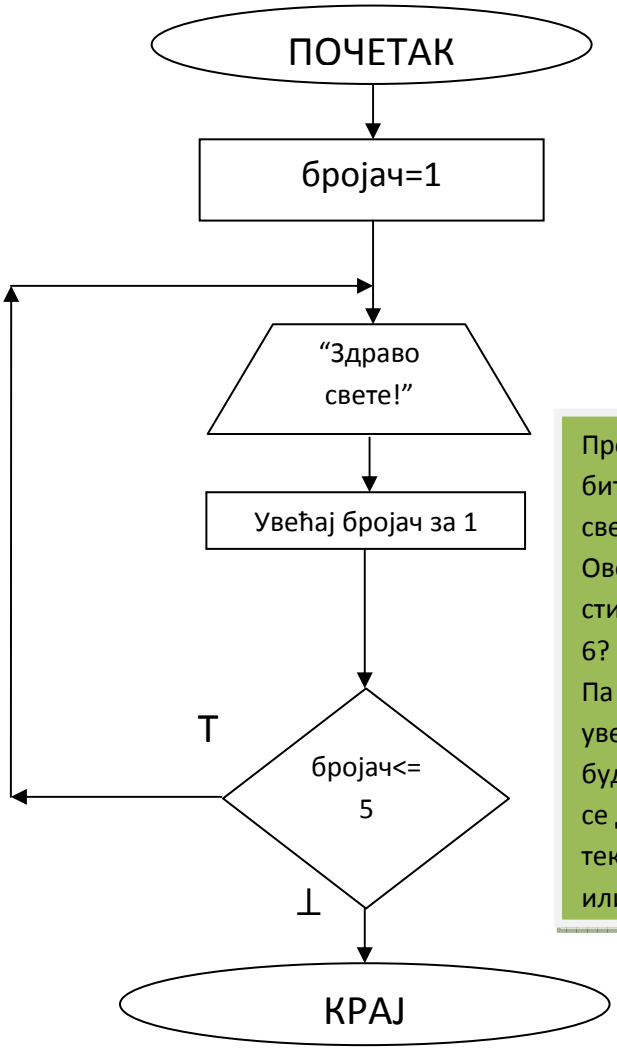
Бројачки циклус ћемо користити у задацима где се од нас тражи да пребројимо број парних бројева, да испишемо првих х природних бројева, итд.

Пример бројачког циклуса: ***Исписати ” Здраво свете!” пет пута.***

Шта се од нас тражи? Да испишемо текст “Здраво Свете!” десет пута.
Овај проблем бисмо могли решити и цртањем исписа са истим текстом
пет пута:



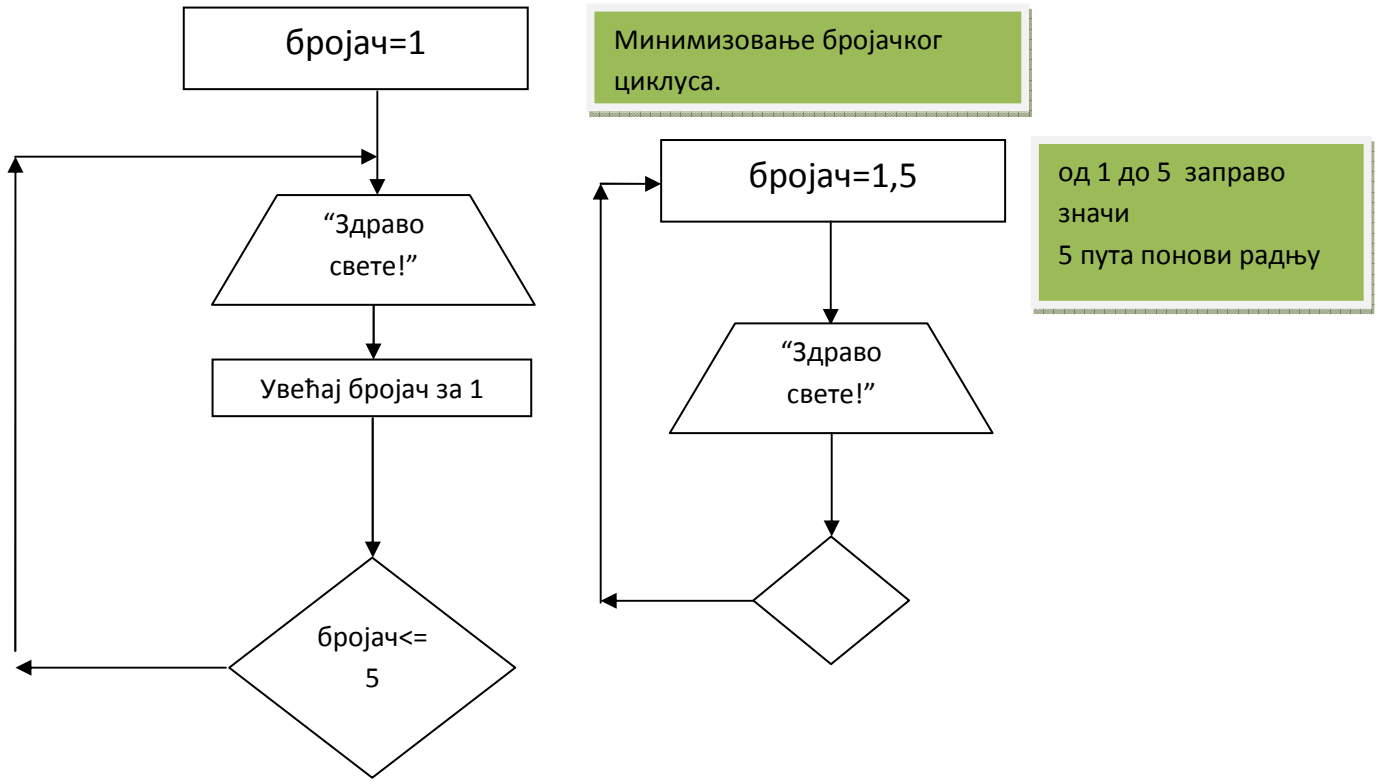
Али да се од нас тражи да испишемо тај текст неколико хиљада пута, овај начин би био бесмислен, тако да ћемо реализовати решење уз помоћ бројачког циклуса. Да би то успели, требаће нам бројач, механизам за његово увећавање и услов написан у складу с њима. Будући да је овај пример сложенији, сваки део ће бити објашњен унутар зелених оквира који нису део алгоритма већ само коментари, ради лакшег разумевања.



Уводимо бројач и додељујемо му вредност 1

Проверава се тачност услова, ако је тачан бројач ће бити увећан за 1 и поново ће се исписати „Здраво свете“
Ове радње ће се извршавати све док бројач не стигне до 6. Зашто ће стати када бројач буде једнак 6?
Па за први пролаз је бројач=1 испише се текст , увећа се бројач за 1 провери тачност услова , будући да 1 јесте мање или једнако од 5 , наставља се даље ....за пети пролаз бројач=5, исписује се текст, увећава се бројач за 1 и како 6 није мање или једнако 5, завршава се циклус.

Будући да се делови бројачког циклуса углавном исти за сваки такав циклус често се представља у другој, минимизованој форми.



Оба дијаграма раде исто, на исти начин, само што је другачије представљено.

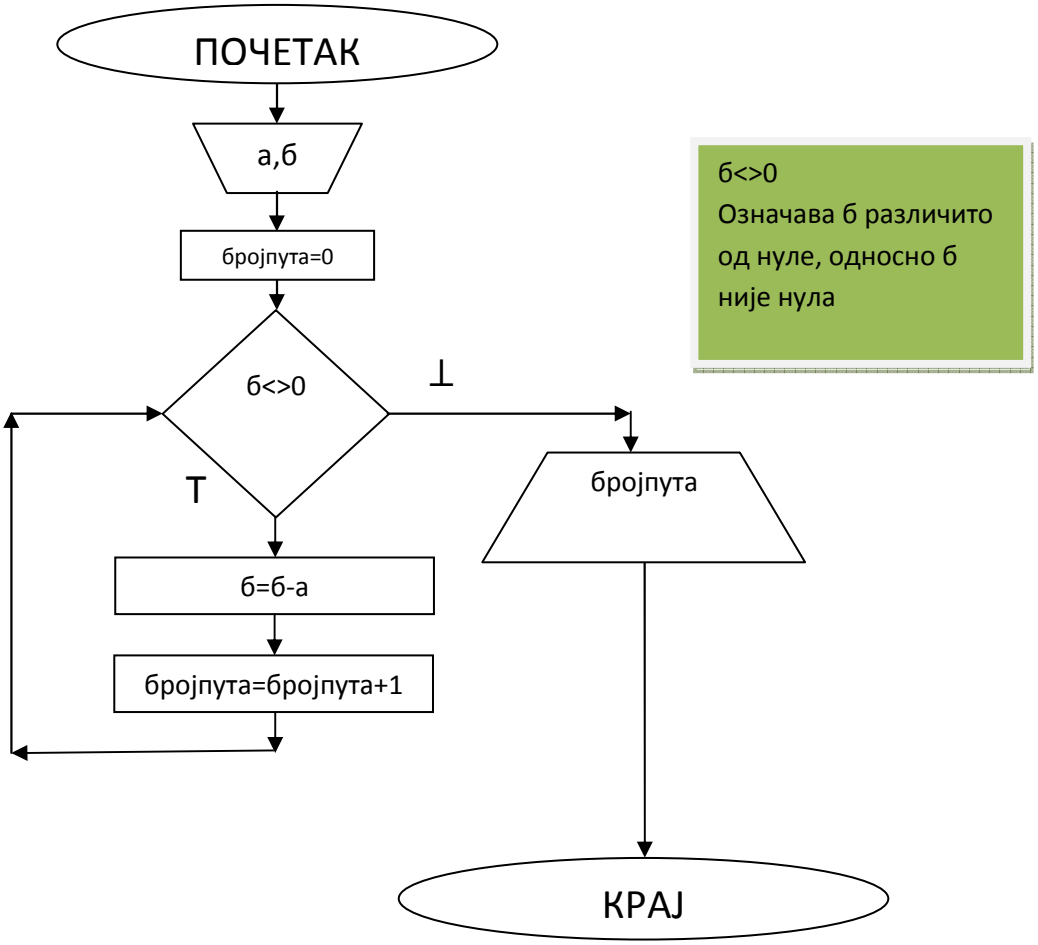
1.3.2 Циклус да предусловом

Пре него што објаснимо овај циклус, морамо нагласити да се сваки циклус може реализовати и преко бројачког. Оно због чега бирамо један уместо другог, јесте разумљивост, односно бирамо онај који нам је природнији. Тако да ако чекамо да се неки услов испуни и не

знамо када ће се то тачно десити , боље је да користимо преусловни циклус.

По контрукцији је врло сличан бројачком, с једином разликом што не зависи од бројача већ неког услова који се може мењати и пре уласка у циклус.

Пример: ***Избројати колико пута се мора одузети унети број а од броја б док б не буде једнако нули.***



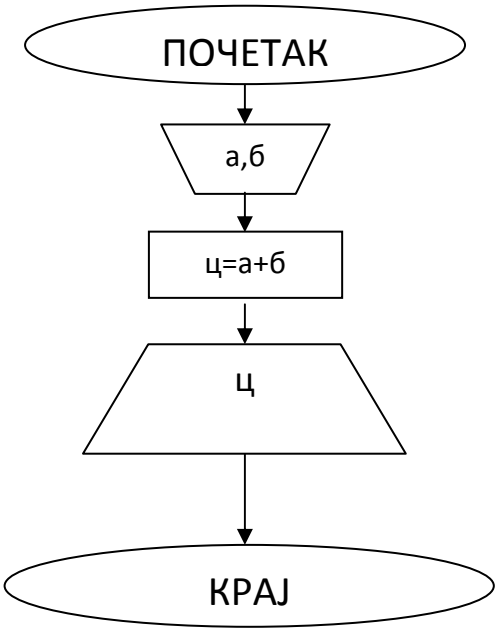
2. Типови алгоритама

-По структури алгоритме можемо поделити на линијске алгоритме, алгоритме који се гранају и цикличне алгоритме.

2.1 Линијски алгоритми

Линијски алгоритми су они алгоритми у којима је цео поступак састављен у **једној континуалној линији**, један поступак испод другог и тако даље.

Пример: Алгоритам који врши сабирање два унета броја и исписује њихов резултат.

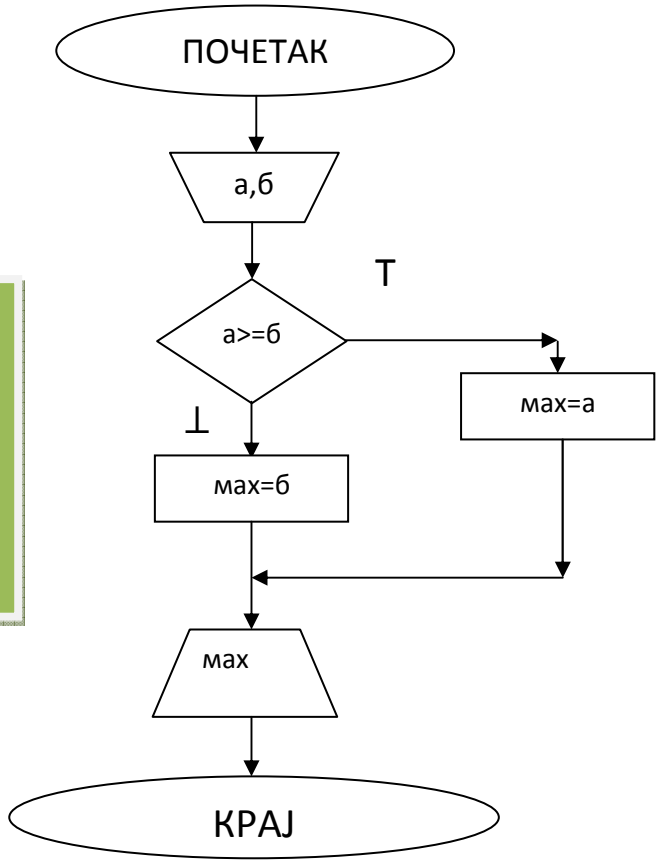


2.2 Алгоритми са гранањем

-Алгоритми који се гранају су они алгоритми који у свом саставу садрже **логички услов**, који може бити тачан или нетачан. Такође, могу се гранати по вредностима.

Пример 1: Алгоритам који налази максимум од 2 унета броја.

Најпре се уносе неке вредности а и б. Следећи корак је упоредити *а* са *б*. Постоји више начина за реализовање поређења, овде је употребљена операција веће или једнако. Ако је *а* веће од *б*, то значи да је *а* максимум

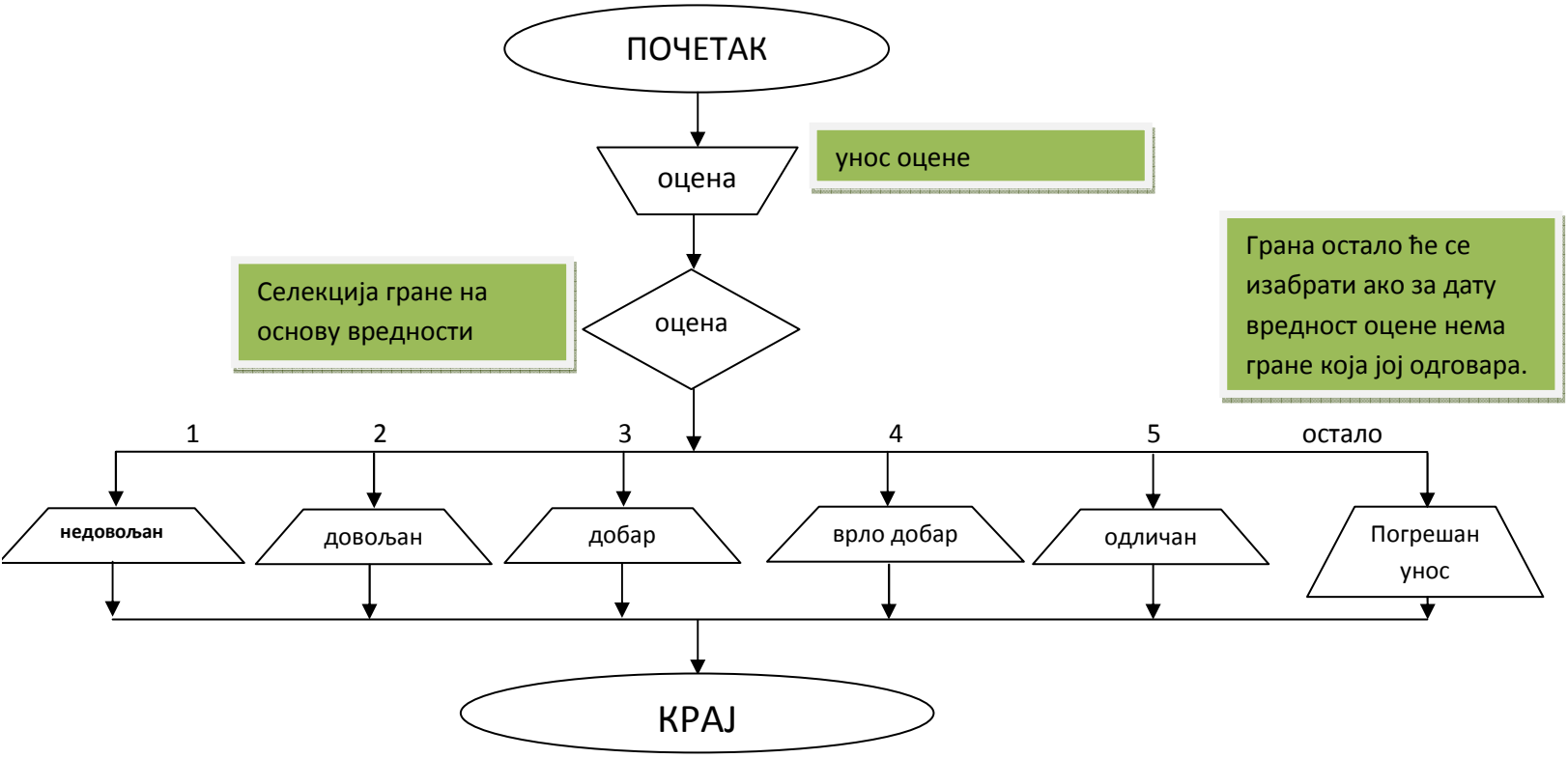


Максимум неког скупа је највећа вредност у том скупу.
А={1,2,2,2,2,3,4,5,5,5,5} је исто што и А={1,2,3,4,5}
Б={1,1}
Max (А)=5
Max (Б)=1

а>=б
Означава а је веће или једнако од б

а<=б би означавало а мање или једнако б

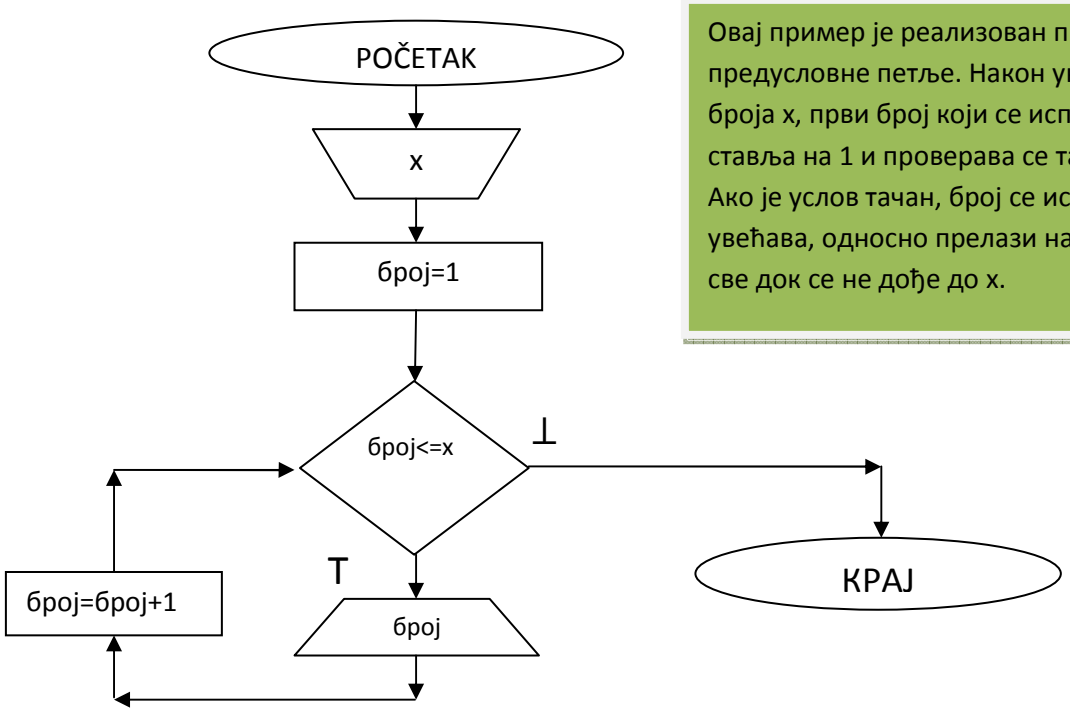
Пример 2: На основу унете оцене ђака исписати одговарајући опис оцене.
5 - Одличан, 4 – Врло добар , 3 – Добар , 2 – довољан, 1 – недовољан



2.3 Циклични алгоритми

Циклични алгоритми имају особину да садрже у себи операције које се понављају, тј. део алгоритма се “врти у круг”.

Пример: Исписати првих x природних бројева.



Овај пример је реализован преко предусловне петље. Након уноса неког броја x, први број који се исписује се ставља на 1 и проверава се тачност услова. Ако је услов тачан, број се исписује и увећава, односно прелази на следећи број, све док се не дође до x.

3. Решени задаци са објашњењем поступка

Сваки задатак ће бити урађен и то са сваким од корака, раније наведеним као стандардним при формирању алгоритма односно решавању проблема.

1. Разумевање проблема

2. Записивање на природном језику идеја за решавање проблема

3. Изградња модела (писање математичких формула, израза)

4. Формулисање алгоритма

Задатак 1

Нацртати алгоритам који исписује растојање између две унете тачке.

Решење:

1 – 2 Разумевање проблема и изградња идеја на природном језику

У задатку се тражи растојање између две унете тачке. Прво што сада сигурно знам јесте да ћу морати имати блок за унос 4 непознате.
4 непознате јер свака тачка има x и y координату, а ми уносимо 2 тачке.
Даље, формула за растојање између две тачке је

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Израчунавање корена ћемо изначавати са sqrt (скраћеница од енглеског израза за квадратни корен односно „square root“) , а дизање неког израза на квадрат са sqr (square) или ^2 .

И још морамо резултат исписати.
Сада смо разумели и смислили решење за проблем. Сада ћемо га мало формалније записати.

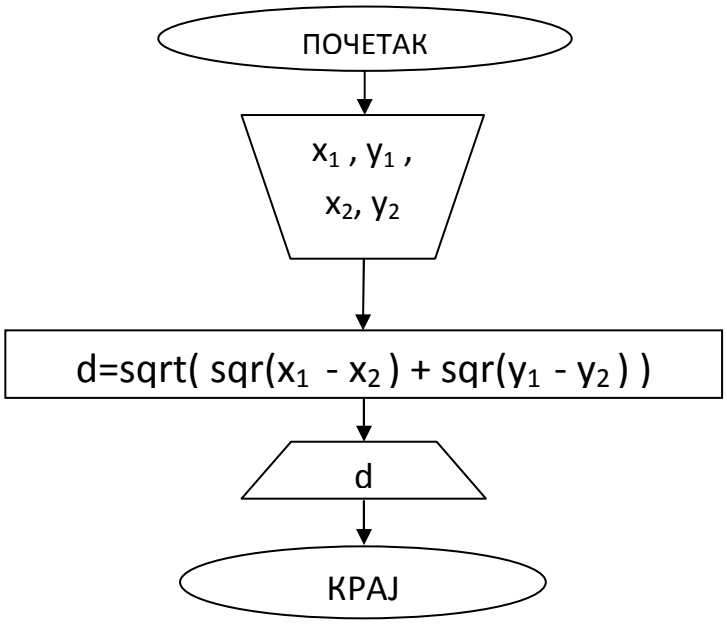
3. Изградња модела (писање математичких формула, израза)

x_1 , y_1 , x_2, y_2

d=sqrt((x_1 - x_2)^2 + (y_1 - y_2)^2)
или
d=sqrt(sqr(x_1 - x_2) + sqr(y_1 - y_2))

испис д

4. Формулисање алгоритма



Задатак 2

Формулисати / нацртати алгоритам који приказује поступак израчунавања нето плате радника по сату. Бруто износ плате радника по минуту је 2 динара, укупан порез који се одбија је 15%

бруто износа плате.

Решење:

1 – 2 Разумевање проблема и изградња идеја на природном језику

-Шта се тражи?

- -Тражи се нето плата радника по сату

-Шта нам је дато?

--Бруто плата радника по минуту и износ који се од ње одбија

-Имам бруто плату у минута , али ми треба у сатима

-- БРУТО= бруто_у_минутима *60

-Како се рачуна нето плата радника?

-- НЕТО= БРУТО – ИЗНОС_који_се_одбија

-Како да израчунам ИЗНОС_који_се_одбија ?

Износ који се одбија је заправо 15% износа бруто плате

Односно износ=БРУТО * (15/100)

- НЕТО= БРУТО – ИЗНОС_који_се_одбија

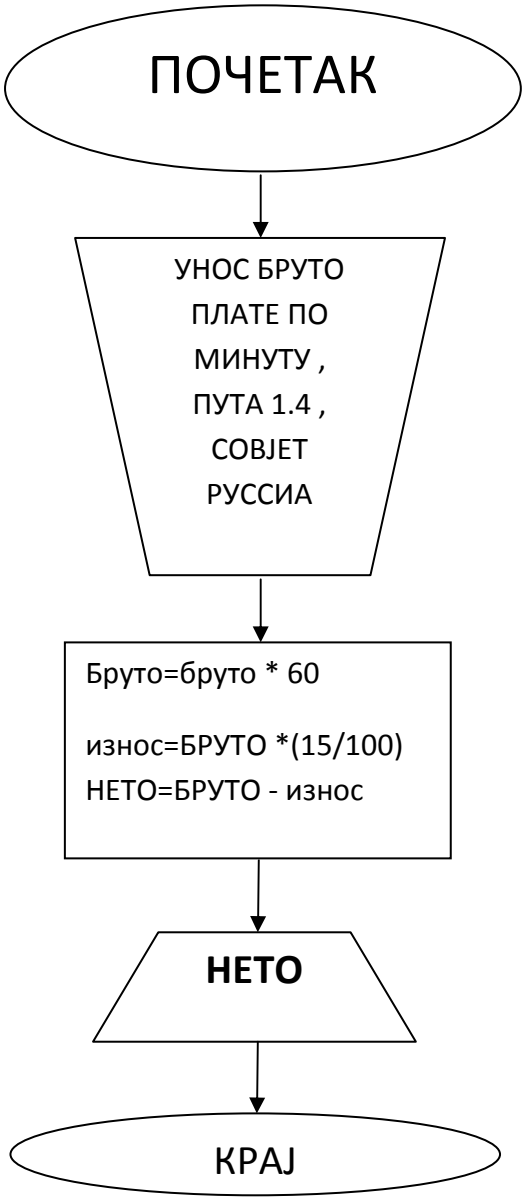
3. Изградња модела (писање математичких формула, израза)

БРУТО= бруто_у_минутима *60

ИЗНОС_који_се_одбија =БРУТО * (15/100)

НЕТО= БРУТО – ИЗНОС_који_се_одбија

4. Формулисање алгоритма



Унутар блока за рачунање се може одвити неограничен број израчунавања вредности. У овом блоку се израчунавају 3 вредности, бруто по сату, износ који се одбија и на крају нето плата.